Nearest prototype classification of noisy data

Fernando Fernández · Pedro Isasi

Abstract Nearest prototype approaches offer a common way to design classifiers. However, when data is noisy, the success of this sort of classifiers depends on some parameters that the designer needs to tune, as the number of prototypes. In this work, we have made a study of the ENPC technique, based on the nearest prototype approach, in noisy datasets. Previous experimentation of this algorithm had shown that it does not require any parameter tuning to obtain good solutions in problems where class limits are well defined, and data is not noisy. In this work, we show that the algorithm is able to obtain solutions with high classification success even when data is noisy. A comparison with optimal (hand made) solutions and other different classification algorithms demonstrates the good performance of the ENPC algorithm in accuracy and number of prototypes as the noise level increases. We have performed experiments in four different datasets, each of them with different characteristics.

Keywords Nearest prototype classification · Evolutionary learning · Machine learning

1 Introduction

The design of nearest prototype classifiers relies on the way of defining the number of prototypes, as well as their position Kuncheva and Bezdek (1998), Seo et al. (2003). It is known that by introducing more and more prototypes, the accuracy over the training set is increased. However, it is also known that over-fitting is very common when a high number of prototypes is used, so a way to balance both training accuracy and over-fitting is required. This problem is worse when data is noisy, because fitting over noisy training data may produce a very bad solution over the test sets.

To solve these problems, classical algorithms Kohonen (1984), Fritzke (1994) are parametrized with several predefined values, such as the number of prototypes to use (or neurons in the neural network bibliography), the initial set of prototypes, how many prototypes are assigned to each class, a smoothing parameter, etc. Typically, the success of the method depends on all these parameters. When the data is noisy, the definition of all these parameters must change because, for instance, it is supposed that a higher adaptation to the training data (typically achieved by increasing the number of prototypes) may produce worse adaptation to the test data. Indeed, it is difficult, without an "a priory" study of the data, to decide when we could obtain a better solution by increasing the number of prototypes, and when it is necessary to reduce that number because we are over-fitting the training data. We must think that in most real problems the amount of noise is not known.

The Evolutionary Design of Nearest Prototype Classifiers (ENPC), whose biological inspirations are described in Fernández and Isasi (2002), offers a solution to the design of the nearest prototype classifiers. It evolves a population of prototypes that are able to correctly classify the training data, maintaining good properties of size and hence, of generalization capabilities. Previous work Fernández and Isasi (2004) has shown that the algorithm produces very good solutions in domains where data is not noisy, and without requiring the number of prototypes, the initial set of prototypes, or a smoothing parameter.

In this work, we present the results of the ENPC algorithm in several noisy domains in order to study the performance of the algorithm when noisy data appears. The algorithm is compared with "hand-made" solutions, as well as with other classifiers obtained from the literature. The first two datasets are the LED display Brieman et al. (1984) and spiral dataset. With this two domains, we can study the performance of several algorithms when the amount of noise varies. Another dataset, Isolet dataset allows us to test the algorithms with high dimensional real data. And last, the Waverform dataset allows us to measure the behaviour of the algorithms when noisy features are introduced in the original data. The next section shows a brief description of the ENPC algorithm, and in Sect. 3 the experiments performed over both domains are shown. Section 4 concludes this paper.

2 The ENPC algorithm

The ENPC algorithm is an evolutionary approach to design nearest prototype classifiers. It solves the initialization problems of nearest prototype approaches by the concept of evolution, that allows the classifier itself to make the operations required to achieve a successful solution. The algorithm follows the paradigm of evolutionary computation, where a population of individuals-solutions evolves by the iterative execution of some genetic operators, toward a near optimal set of solutions. In this case, the individuals are the prototypes, the population size is not constant, and the solution searched for is not an individual but the whole population. The population constitutes the set of prototypes used for classification using the nearest neighbor rule.

The algorithm begins with an initialization where the classifier is composed of only one prototype. Then, the classifier evolves by executing, in a loop, all the designed operators described below (as shown in Fig. 1).

These operators take advantage of some information gathered at the beginning of the loop. The classification accuracy of the prototype is the percentage of instances located in the region of a prototype and that are classified correctly. Basically, if the prototype is of a given class, the accuracy of the prototype is the number of instances of such class divided by the total number of instances located in the

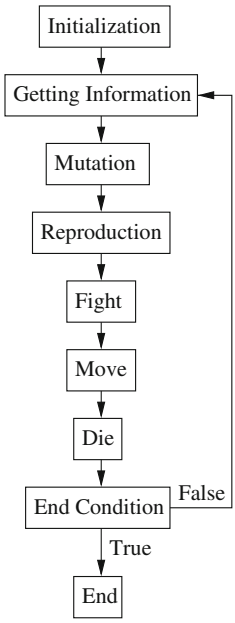

Fig. 1 ENPC algorithm flow

region of the prototype. The coverage of a prototype is the ratio between the number of instances that it classifies (tha are located in its region) and the total number of instances in the dataset. Lastly, the quality of a prototype, is a function of the classification accuracy of the prototype and the coverage of the prototype. An extensive definition and formal description of accuracy, coverage and quality can be found in a previous work Fernández and Isasi (2004). Next, we describe the different operators.

2.1 Initialization

One relevant feature of this method is that the number of prototypes, the initial set of prototypes, or a smoothing parameter must not be defined a priori. This is due to three main issues; firstly, the initial number of prototypes is always one, and its initial location is not relevant (it clusters all the domain, wherever it is located); secondly, the method is able to generate new prototypes stabilizing in the most appropriate number; and last, there is no smoothing parameter, given that the method automatically adjusts the intensity of change in prototypes taking into account their qualities in each iteration.

2.2 Mutation operator

A region is defined by each prototype following the nearest neighbor rule. Therefore, the goal of this operator is to label each prototype with the most popular class in each region. Each prototype knows the number of patterns of each class located in its region. Then, the prototype changes, if needed, and becomes the same class as the most abundant class of

patterns in its region. This operation maximizes the local accuracy of the prototype given the instances located in its region.

This way to obtain the main class is typically used when unsupervised learning is applied to supervised classification Bermejo and Cabestany (2000), Pal et al. (1993), and is typically called the labelling phase.

2.3 Reproduction operator

The goal of this operator is to introduce new prototypes in the classifier. The insertion of new prototypes is a decision that is taken by each prototype, in the sense that each prototype has the opportunity of introducing a new prototype in order to increase its own quality. Therefore, the prototype will insert new prototypes when it detects that there are many instances of different classes in its region, i.e. when the local accuracy is low. The goal of this operator is also to increase the local accuracy of the prototype.

2.4 Fight operator

This operator provides the prototype with the capability of obtaining patterns from other regions. The steps to execute are defined as follows:

1. Each prototype, r_i chooses the prototype $r_{i'}$ against which to fight. Prototypes are chosen from the set of prototypes in its neighborhood. To decide which prototype to choose from the neighbors set, a roulette is used assigning to each region $r_j \in neighbors(r_i)$ a slice of size proportional to the difference between its quality and the quality of r_i.
2. Decide whether to fight or not. The probability of fighting between prototypes r_i and $r_{i'}$ is proportional to the distance of their qualities.
3. If prototype r_i decides to fight against prototype $r_{i'}$, there are two possibilities. Given s_i as the class associated to r_i, and $s_{i'}$ the class associated to $r_{i'}$:
   - If $s_i \neq s_{i'}$ (cooperation). Both prototypes belong to different classes. In this case, the prototype $r_{i'}$ will give the prototype r_i the patterns of the class s_i.
   - If $s_i = s_{i'}$ (competition). The winner is decided again by using a roulette with only two slices, each slice belonging to each prototype, and sizes proportional to the qualities of each prototype. Furthermore, the amount of patterns that are transferred depends on a probability proportional to the qualities of both prototypes.

2.5 Move operator

The move operation relocates each prototype in the best expected place. This is the centroid of all the training data or instances in its region that belongs to its same class.

This operation, based on the second step of Lloyd iteration Lloyd (1982), allows us to make a local optimization of each prototype, increasing the performance of the whole classifier.

2.6 Die operator

The probability for a prototype of being eliminated is 1 minus the double of its quality. In that case, successful prototypes will survive with probability of 1, while useless prototypes with quality less than 0.5 might die. A wide range of heuristics about how to reduce the number of prototypes of the classifier can be found Fritzke (1994), Patanè and Russo (2001).

2.7 End condition and classifier selection

Following the idea of introducing a minimal number of parameters, the end condition is fixed to a maximum number of iterations large enough to ensure that a good solution is obtained. Furthermore, given that the algorithm is very fast (each of the executions of the algorithm performed in the experiments below does not take more than 1 min), a high number of iterations is not a time restriction.

Lastly, to choose the right classifier, several methods can be followed from the set of classifiers generated in each of the iterations Fernández and Isasi (2004). In this case, a validation set is used, and the classifier more adapted to this set is then chosen.

3 Experiments

The goal of this paper is to show the performance of the ENPC classifier in problems with noisy data. It should be noted that no additional properties nor parameters must be tuned for the ENPC method. Better, its evolution capability allows it to adapt to different domains and noise situations.

In the community, there exists a lot of benchmark datasets that can be used to evaluate new algorithms. We are interested in domains with noisy data, so we have selected 4 datasets with different objectives. Firstly, the spiral dataset, that consists of data following two interlaced spirals, each of them belonging to a different class. Secondly, the LED digit classification problem Brieman et al. (1984) obtained from the UCI Machine Learning repository Blake and Merz (1998). Both of them allow us to introduce different noise levels in the data in order to study the relationship between the noise and the classifier performance. Then, the Isolet dataset from UCI is used to test the performance of the algorithm over real high-dimensional data; and lastly, the Waveform dataset, also extracted from UCI, has been used to test the algorithm over data with additional randomly generated features.[1]

3.1 Spiral dataset

Spiral dataset is typically used in the literature as a challenge set, where data from two interlaced spirals must be correctly classified, as is shown in Fig. 2. There are two interlaced spirals, each one with 500 examples, and examples in the same spiral belong to the same class. Tests are performed over a different test set of the same size.

This domain is easily corrupted by noise by randomly moving each example a distance that follows a uniform distribution in the range $[-z, +z]$. For instance, if $z = 150$, the previous dataset is transformed as shown in Fig. 3, where it can be appreciated that the two spiral distributions have been destroyed by the noise, forming almost a random distribution of points.

In this work, that noise component has ranged from 0 to 250 (note that points range from -2000 to 2000), and several experiments have been performed in order to compare the ENPC algorithm with other classification algorithms, such as decision trees (J48) Quinlan (1993), decision rules (PART) Frank and Witten (1998), Naive Bayes Duda and Hart (1973), and IBK Aha and Kibler (1991), for values of $k = 1$ and $k = 3$, all of them executed in

[1] We have selected four domains that permit to study the behavior of the ENPC algorithm in depth, reporting a wide study over them. Experiments in other classical domains like Iris, Pima Indian Diabetes, etc. can be found in Fernández and Isasi (2004). However, we do not consider them relevant to the topic we are studying in this manuscript.

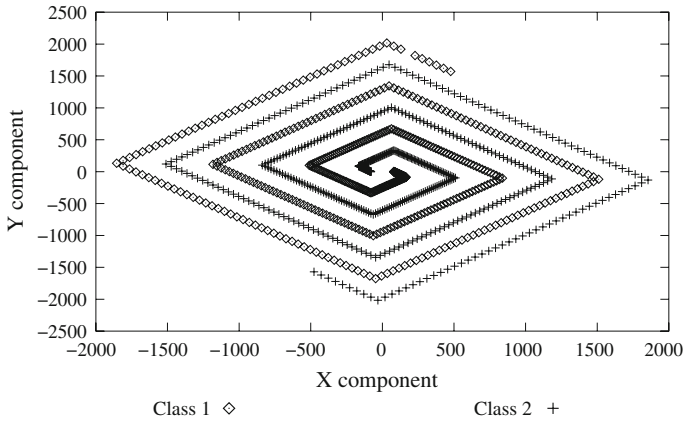

Fig. 2 Spiral dataset without noise

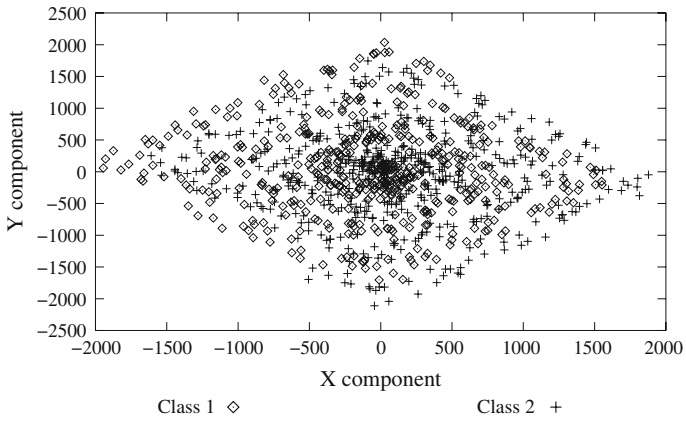

Fig. 3 Spiral dataset with noise

WEKA Witten and Frank (2000) with the predefined parameters. Furthermore, an additional hand-made solution is proposed to be a 1 nearest neighbor approach that uses the whole training set with noise 0 (shown in Fig. 2) as the set of prototypes, and which is supposed to return results very close to the optimal Bayesian solution. Experiments have been performed using a training set of 1,000 instances, and a test set of the same size. All the experiments are summarized in Fig. 4.

As it was expected, the best approach is obtained by the hand-made solution. This approach returns a 100% rate of success until the noise level achieves a value of 90. This is because until that moment, instances from one spiral do not mix with instances from the other spiral. From that noise level, the instances of the different spirals begin to mix, and they can not be distinguished with the Bayesian nor the nearest neighbor algorithms, obtaining 58.5% classification success when the noise level is 250.

The figure shows that the worse approach is, in this case, Naive Bayes, which is not able to learn anything even when the data has not noise. J48 and PART obtain similar results. When the data noise level is lower than 90, they obtain successes of around 70%. However, when the data of the different spirals mixes, the success decreases to almost 50%. IBK, both for

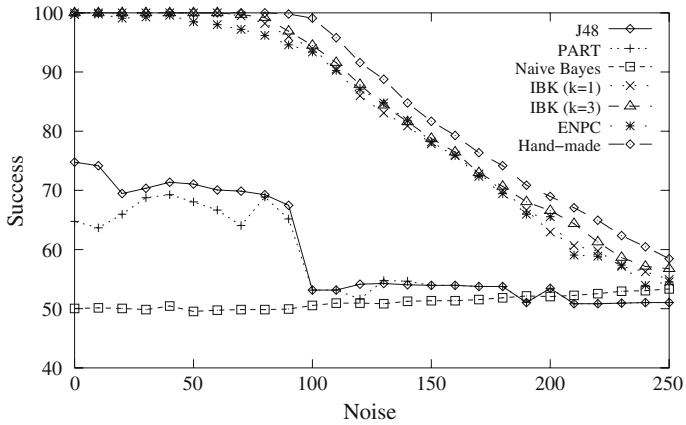

Fig. 4 Performance of several algotithms over the spiral dataset

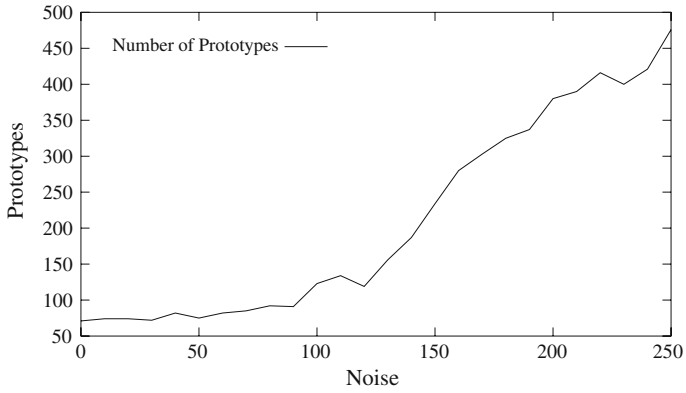

Fig. 5 Number of prototypes achieved by ENPC in the spiral dataset

$k = 1$ and $k = 3$, obtains very good results, even with high noise levels, and they are always very close to the hand-made approach. Lastly, ENPC returns similar results to IBK, showing that with high noise levels it achieves very good results, with a reduced set of prototypes.

The number of prototypes obtained by ENPC for each noise level is shown in Fig. 5. The figure shows that while the noise level increases, the number of prototypes also increases. However, this increment is larger when data of the two spirals begin to mix (with a noise level of 90) so it is more difficult to separate data from noise. The algorithm tries to improve the training performance by increasing the number of prototypes, and hence, fitting the training set. Notice that the extreme case is the 1-nearest neighbor approach ($k = 1$) with the whole dataset used for classification. However, it also has been shown that this increment in the number of prototypes does not keep us from obtaining good results for test data.

3.2 LED digit classification

The LED digit classification problem consists of recognizing which of the 10 digits is being displayed on a LED. For each segment of the LED, the values are supposed to be 1 if the segment is on, and 0 if the segment is off. Furthermore, each segment of the LED is supposed

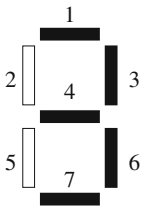

Fig. 6 LED representing the number 3

to fail $p\%$ of the times, so the value for a segment can be 1, even when the segment is off, and vice versa. From a nearest neighbor point of view, this problem has a theoretical optimal solution of 10 prototypes, each one for a different class. For instance, Fig. 6 shows the values of the prototype representing value 3 in the LED. The value of the prototype representing this class in the nearest prototype classifier should be $\{1, 0, 1, 1, 0, 1, 1\}$.

We have generated two sets of 1,000 examples, one for training and the other for testing. Furthermore, the first set has been divided again in two sets of 500 examples. The first one can be considered the training set, given that it will contain the data used by the ENPC algorithm to learn. The second sub-set can be considered as a validation set that will be used to choose the final classifier from the set of classifiers generated by the ENPC algorithm in each of its iterations.

Figure 7 shows the evolution of a set of prototypes during 300 iterations of the ENPC algorithm over the 500 training and validating examples. In this case, the data is noisy with a value of 10%. The figure shows that in only 13 iterations, the algorithm is able to achieve a set of 10 prototypes, with a classification accuracy over the training set of 67%, the same as for the validation set. Given that 300 iterations are executed, the algorithm searches for better solutions, looking for solutions around 10 prototypes. The best solution is obtained in iteration 282, with an accuracy of 74.4% for the training set, and 76.0% over the validation set. This classifier is chosen for passing the test, achieving a success of 72.9% over the 1,000 test examples. For comparisons, we also have executed a test over the optimal set of prototypes composed of 10 prototypes defining each of the numbers. This classifier achieves a performance of 74.5% over the test set, only 1.6% more than the solution achieved by the ENPC algorithm. The optimal Bayes classification rate is 74% (which denotes that test set is not fully representative, but enough for demonstrating our goals).

The experiment described above has been executed for each value of noise, from 0 up to 20%. For each of these values of noise, the ENPC algorithm has been executed 20 times, given that the algorithm has a strong stochastic component. From the total of 2,000 examples, 500 have been used for training, 500 examples for validating and choosing the final classifier, and 1,000 examples for testing. For the rest of algorithms, the first 1000 examples were used for training and the other 1,000 for testing. Furthermore, the test has been executed over the optimal (hand-made) classifier described above. Figure 8 shows a summarization of these experiments.

When there is not noise, we can see that the ENPC classifier optains the optimal solution of a set of 10 prototypes with a success for test data of 100% in each of the 20 executions performed. When the noise parameter is increased, the success decreases for the classifiers obtained with ENPC as well as with the optimal classifier. However, both values stay very close, independently of the noise level.

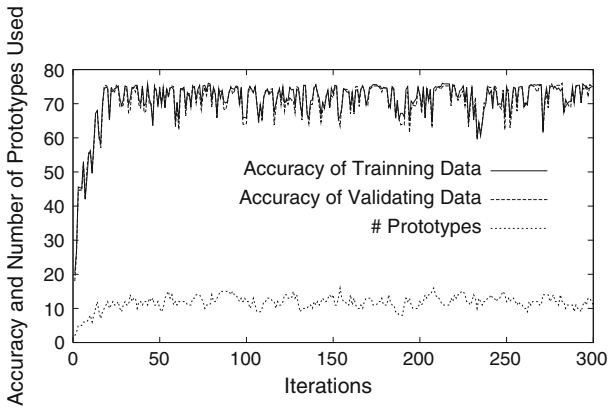

Fig. 7 Evolution of the ENPC algorithm over the LED display dataset

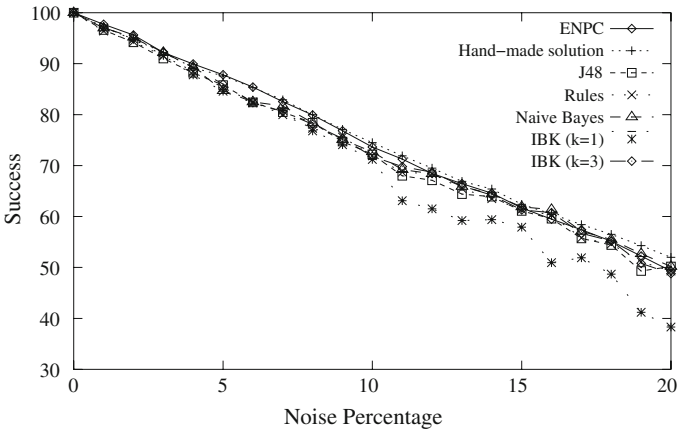

Fig. 8 Performance obtained by different algorithms over the LED display dataset

The rest of algorithms achieve similar results, except for KNN with $k = 1$, where overfitting problems appear, reducing its performance, and showing the difficulty of choosing a right k parameter of this approach, and how it may depend on the domain and the noise level.

The generalization capabilities of the ENPC classifier can be understood if we take a look at the evolution of the method in the number of prototypes, shown in Fig. 9. When noise is 11%, the average number of prototypes achieved in the 20 executions of the algorithm is still lower than 11 (10.67). An average of 12 prototypes is achieved when noise is 16%, and an average value of 13 is achieved with a noise of 18%. Furthermore, note that the variance of the number of prototypes is lower than 1 when the value used for the noise is lower than 12%, showing that in the different executions, the number of prototypes stays similar.

3.3 Isolet dataset

The Isolet Spoken Letter Recognition dataset, extracted also from UCI Machine Learning Repository, consists of 7,788 instances, each of them belonging to a class from a set of 26 classes (one for each letter). Each instance is featured by 617 attributes, all of them real-

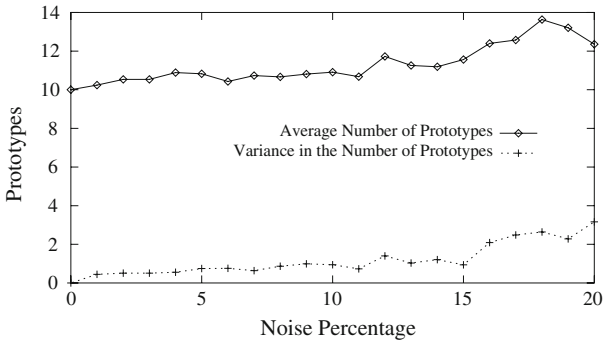

Fig. 9 Evolution of the number of prototypes in a execution of the ENPC algorithm over the LED display dataset

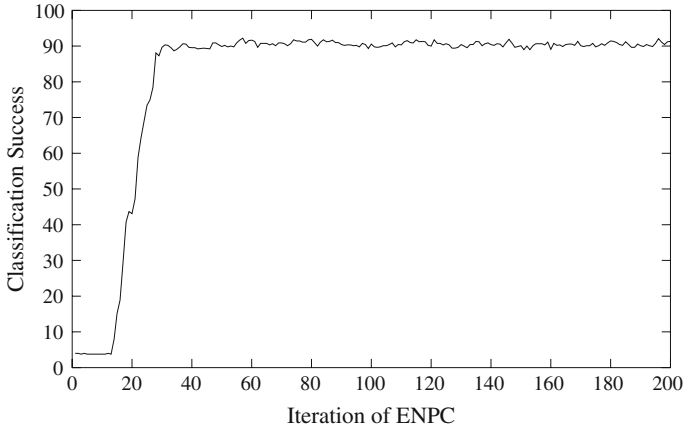

Fig. 10 Evolution of the classification success on training when executing ENPC over Isolet dataset

valued scaled into the range −1.0 to 1.0. Given the high number of attributes and classes, and that data was obtained from real speakers, it is considered very useful for evaluating algorithms' performance in noisy domains.

Figure 10 shows the evolution of the classification accuracy over the training set of an execution of the ENPC algorithm over the Isolet domain. The figure shows that in less than 40 iterations, the algorithm is able to obtain a 90% of success over the training set, maintaining that value until the execution ends after 200 iterations.

Figure 11 shows the evolution of the number of prototypes in the same execution of the algorithm. The number of prototypes grows until the range of 30–35 in also 40 iterations, showing the relationship between the classification accuracy and the number of prototypes. This figure also shows that the algorithm does not overfit training data, given that the relationship between the number of prototypes generated by the algorirthm and the number of classes is very low (less than 2 prototypes per class).

Table 1 shows the results of several different algorithms extracted from the literature. The first column defines the algorithm, the second one defines the accuracy, and the third one the source of the information.

The experimentation has been executed with decision trees (J48) Quinlan (1993), decision rules (PART) Frank and Witten (1998), Naive Bayes Duda and Hart (1973), John and Langley (1992), and IBK Aha and Kibler (1991) for different values of k. The implementa-

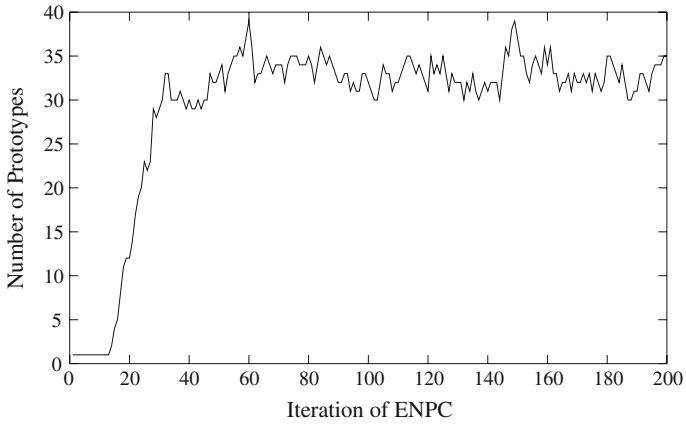

Fig. 11 Evolution of the number of prototypes when executing ENPC over Isolet dataset

Table 1 Comparative results of different algorithms in the Isolet dataset

Algorithm	Accuracy	Source
Naive Bayes	84.4	Ours (WEKA)
SMO	96.4	Ours (WEKA)
IBK ($k = 1$)	87.7	Ours (WEKA)
IBK ($k = 3$)	79.1	Ours (WEKA)
IBK ($k = 5$)	81.4	Ours (WEKA)
J48	80.2	Ours (WEKA)
PART	70.1	Ours (WEKA)
Relief-F Kira and Rendell (1992)	84.6	Wettschereck et al. (1997)
$k - NN_{VSM}$ Wettschereck (1995)	86.1	Wettschereck et al. (1997)
CCF Creecy et al. (1992)	83.1	Wettschereck et al. (1997)
VDM Stanfill and Waltz (1986)	80.3	Wettschereck et al. (1997)
MVDM Cost and Salzberg (1993)	85.8	Wettschereck et al. (1997)
MI Shannon (1948), Wettschereck (1995)	85.8	Wettschereck et al. (1997)
ENPC	86.0	Ours

tion of these algorithms is provided by WEKA Witten and Frank (2000), and they are used with the pre-defined parameters. For all of the algorithms, a 10 fold cross validation has been executed. The method has also been compared with different nearest neighbour approaches that allow feature selection and weighting included in Wettschereck et al. (1997).

The table shows the best result for SMO, with a 96.4% rate of success. IBK obtains good results or not depending on the k parameter, from 79.1% for $k = 3$ up to a 87.7% for $k = 1$. This shows the dependency of IBK with the k parameter, given that in the LED display dataset, the best value for k was 3 (the worst value for this dataset). ENPC obtains 86.0% without tuning any parameter, showing a very good performance, higher than most of the other algorithms tested.

Table 2 Comparative results on the waveform-21 and waveform-40 datasets

Algorithm	Waveform-21	Waveform-40	Source
Naive Bayes	82.5	79.1	Our (WEKA)
SMO	85.6	84.8	Our (WEKA)
IBK ($k = 1$)	77.1	73.5	Our (WEKA)
IBK ($k = 3$)	78.5	77.8	Our (WEKA)
IBK ($k = 5$)	79.3	78.1	Our (WEKA)
J48	73.1	74.6	Our (WEKA)
PART	75.0	73.0	Our (WEKA)
Relief-F Kira and Rendell (1992)	82.4	69.78	Wettschereck et al. (1997)
$K - NN_{VSM}$ Wettschereck (1995)	81.6	71.61	Wettschereck et al. (1997)
CCF (cross category feature importance) Creecy et al. (1992)	76.0	77.9	Wettschereck et al. (1997)
VDM (value diferenced metric) Stanfill and Waltz (1986)	78.4	80.6	Wettschereck et al. (1997)
MVDM Cost and Salzberg (1993)	78.2	80.9	Wettschereck et al. (1997)
MI (mutual information) Shannon (1948), Wettschereck (1995)	82.6	82.3	Wettschereck et al. (1997)
ENPC	82.0	83.4	Our

3.4 Waveform

This domain has also been obtained from the UCI Machine Learning Repository Blake and Merz (1998). The first version of this domain, called Waveform-21, consists of 21 relevant features to discriminte 3 different classes. The 21 features are continuous. The second version adds another 19 irrelevant features. So, this domain is very interesting to verify the behaviour of the algorithms when useless features are present. The dataset consists of 1,000 data, and results are obtained from a 10-fold cross-validation. Table 2 summarizes the results.

From the results, several conclusions can be obtained:

- SMO obtains the best results, both for Waveform-21 and Waveform-40.
- Naive Bayes is influenced by the 19 more irrelevant features included in Waveform-40, decreasing its performance by around 3 points
- IBK obtains results under 80% in all the cases, and the influence of the irrelevant features is stronger for lower values of k.
- Decision trees and rules (J48 and PART, respectively) also obtain poor results, but they do not seem to be very influenced by the irrelevant features.
- Both Relief-F and $K - NN_{VSM}$ obtain good results for Waveform-21, but their performance decreases in around 10 points when irrelevant featurs are included.
- CCF, VDM, MVDM and MI obtain similar results for both datasets, showing that their capability to manage irrelevant features is higher.
- ENPC obtains very good results both in Waveform-21 and in Waveform-40, only improved by MI and SMO.

Therefore, it is demonstrated again that ENPC obtains very good results without requiring to define initial parameters.

4 Conclusions

Distinguishing between useful data and noise is a difficult task. The complexity of a good classification grows as the noise increases. The ENPC technique shows a good behaviour when dealing with a noisy domain, and obtains such good result because of two main reasons. On one hand, the number of prototypes is increased only when required. So, when working with noise free domains, it dramatically reduces the number of prototypes achieving values very close to optimal ones from the nearest prototype point of view. When the noise increases, ENPC automatically increases the number of prototypes used, while maintaining this value as low as possible to avoid over-fitting. Thus, the accuracy of the algorithm is very good independently of the noise level.

On the other hand, no parameter nor initial condition needs to be tuned in order to achieve good results. No parameter has been modified in all the experiments performed in this work, except the number of iterations. And this is critical when dealing with problems where the level of noise can not be known in advance. In this case, ENPC is able to automatically adjust internally to the noise, producing good levels of prediction, independent of the characteristics of the domain.

The accuracy of the method reaches a good level in the four domains and it is placed between the best solutions found, not always being the best alternative, but improving all the other methods in some domain.

We can conclude that ENPC is a good method for the supervised classification problem in noisy domains, both in accuracy and proximity to the optimal solutions. No parameter is required, the achieved solutions for each run are stable (the variance of all the solutions given by the method is very low when experiments are executed several times) and the method is able to automatically provide the number of prototypes needed.

Acknowledgements This work has been partially supported by the spanish MICIIN project TIN2008-06701-C03-03, and the regional project CCG08-UC3M/TIC-4141

References

Aha D, Kibler K (1991) Instance-based learning algorithms. Mach Learn 6:37–66

Bermejo S, Cabestany J (2000) A batch learning algorithm vector quantization algorithm for nearest neighbour classification. Neural Process Lett 11:173–184

Blake CL, Merz CJ (1998) UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html

Brieman L, Friedman JH, Olshen RA, Stone CH (1984) Classification and regression trees. Wadsworth, London

Cost S, Salzberg S (1993) A weighted nearest neighbour algorithm for learning with symbolic features. Mach Learn 10:57–78

Creecy RH, Masand BM, Smith SJ, Waltz DL (1992) Trading mips and memory for knowledge engineering. Communications of the ACM 35

Duda RO, Hart PE (1973) Pattern classification and scene analysis. Wiley, New York

Fernández F, Isasi P (2002) Automatic finding of good classifiers following a biologically inspired metaphor. Comput Inform 21(3):205–220

Fernández F, Isasi P (2004) Evolutionary design of nearest prototype classifiers. J Heuristics 10(4):431–454

Frank E, Witten IH (1998) Generating accurate rule sets without global optimization. In: Proceedings of the fifteenth international conference on machine learning

Fritzke B (1994) Growing cell structures -a self-organizing network for unsupervised and supervised learning. Neural Networks 7(9):1441–1460

John GH, Langley P (1992) Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the Eleventh conference on uncertainty in artificial intelligence, pp 338–345

Kira K, Rendell LA (1992) A practical approach to feature selection. In: Proceedings of the Ninth international conference on machine learning. Morgan Kaufmann, pp 249–256
Kohonen T (1984) Self-organization and associative memory, 3rd edn. Springer, Berlin Heidelberg 1989
Kuncheva LI, Bezdek JC (1998) Nearest prototype classification: clustering, genetic algorithms, or random search. IEEE Trans Syst Man Cyber 28(1):160–164
Lloyd SP (1982) Least squares quantization in PCM. In: IEEE transactions on information theory, no. 28 in IT, pp 127–135
Pal NR, Bezdek JC, Tsao EC-K (1993) Generalized clustering networks and kohonen's self-organizing scheme. IEEE Trans Neural Networks 4(4):549–557
Patanè G, Russo M (2001) The enhanced LBG algorithm. Neural Networks 14:1219–1237
Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Mateo, CA
Seo S, Bode M, Obermayer K (2003) Soft nearest prototype classification. IEEE Trans Neural Networks 14(2):390–398
Shannon CE (1948) A mathematical theory of communication. Bell Syst Technol J 27:379–423
Stanfill C, Waltz D (1986) Toward memory-based reasoning. Commun Assoc Comput Mach 29:1213–1228
Wettschereck D (1995) A description of the mutual information approach and the variable similarity metric. Tech. Rep. 944, German National Research Center for Computer Science, Artificial intelligence research division, Sankt Augustin, Germany
Wettschereck D, Aha D, Mohri T (1997) A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artif Intell Rev 11:273–314
Witten IH, Frank E (2000) Data mining. Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, London